Learning Highcharts 4

Highcharts is a popular type of web charting software that produces stunning and smooth animated HTML5 SVG graphs.

Joe Kuan continues the Learning Highcharts series, with the second edition comprising of brand new content and revised chapters. The book covers all the graphs supported in Highcharts 4. Step-by-step tutorials demonstrate how to create presentable charts with real-life data. The book also explores how to replicate industry charts from a financial brochure, sports article, 3D infographics, and even a Fiat 500 speedometer.

The second part of the book focuses on the operational side; APIs, events, and framework integration. The third part of the book looks into the server side and online services, how Highcharts/PhantomJS works, and demonstrates the usage and reviews the latest exciting development, Highcharts Cloud—an online chart service without any prior Highcharts experience needed. You just need to insert the data and it does the rest.

In a nutshell, this book aims to be the most comprehensive Highcharts book in the market.

Who this book is written for

The book is aimed at all levels of readers. Beginners can learn the basic configurations and step-by-step approaches in creating charts or Highcharts cloud. For intermediate and advanced readers, the book explores the APIs, events, server-side operations and plugins.

What you will learn from this book

- Use Highcharts to create a simple chart and all the graph series supported in Highcharts 4
- Create presentable industry-scale charts with series-specific options
- Get to grips with 3D charts and create special effects with 3D-specific options
- Use the Highcharts APIs and events to produce interactive charts
- Integrate Highcharts with jQuery Mobile and ExtJS's extensions
- Discover how to use Highcharts on the server side and get to know about the Highcharts cloud service
- Write and apply Highcharts plugins to come up with innovative charts

In this package, you will find:

- The author biography
- A preview chapter from the book, Chapter 1 'Web Charts'
- A synopsis of the book’s content
- More information on Learning Highcharts 4

About the Author

Joe Kuan was born in Hong Kong and continued his education in the UK from secondary school to university. He studied computer science at the University of Southampton for his BSc and PhD. After finishing his education, he worked with different technologies and industries in the UK. Currently, he is working for iTrinegy—a company specializing in network emulation, profiling, and performance monitoring. He enjoys working on a variety of projects and applying different programming languages. Part of his role is to develop frontend data and present complex network data in stylish and interactive charts. He has adopted Highcharts in his company products since the early version 2. Since then, he has been contributing blogs (joekuan.wordpress.com) and software (joekuan.org) on Highcharts, Highstocks, and Highmaps. In December 2012, he published his first book, Learning Highcharts, Packt Publishing, which is a comprehensive book on Highcharts covering tutorials, examples, tricks, and tips.

Acknowledgments

When I wrote my first ever book, it was a draining 9 months to grind through alongside my demanding work and spending little time with my family. A year later, Packt Publishing approached me to write the second edition. I was in two minds about going through the hectic schedule again. Since then, Highcharts has become a more complete and improved product and it would be a closure for me to see the book end the same way. The first edition was tough but the new experience kept me motivated towards the goal. This journey has taught me one thing: what it takes and means to keep up the level of momentum.
I would like to dedicate this book to my eldest sister, Valamie, who is courageously fighting cancer. Although she is small, we call her "big sister." She never complains and always puts the interests of other people, especially our parents, above her own, despite what she is going through. Special thanks to my wife for her support and patience again and my boy, Ivan, for understanding why his dad has been glued to the computer all this time.

I would also like to thank Adrian Raposo from Packt Publishing for putting up with me and injecting new spark into this book.

"Be studious in your profession, and you will be learned. Be industrious and frugal, and you will be rich. Be sober and temperate, and you will be healthy. Be in general virtuous, and you will be happy. At least you will, by such conduct, stand the be."

—Benjamin Franklin
Learning Highcharts 4

*Learning Highcharts 4* aims to be the missing manual for Highcharts from every angle. It is written for web developers who would like to learn about Highcharts. This book has the following features:

- It is a step-by-step guide on building presentable charts from basic looking ones
- There are plenty of examples with real data covering all the Highcharts series types—line/spline, column, pie, donut, scatter, bubble, area range, column range, gauge, solid gauge, pyramid, box plot, spider, waterfall, funnel, error bar, heatmaps, and 3D charts
- Subject areas are included that haven't yet been covered in online reference manuals and demos, such as chart layout structure, color shading, and in-depth explanations of some specific options such as sizeBy in the bubble series, groupZPadding in the column series, and how to modify or create plugins
- Applications demonstrating how to create dynamic and interactive charts using Highcharts APIs and events handling are covered
- Applications demonstrating how to integrate Highcharts with a mobile framework such as jQuery Mobile and a Rich Internet Application framework such as Ext JS are also covered
- Applications demonstrating how to run Highcharts on the server side for automating charts generation and exporting their graphical outputs are also covered
- Using the latest online service Highcharts Cloud, you'll learn to embed the graphs into documents
- You'll also learn the structure of Highcharts plugins and how to create plugins

This book is not a reference manual as the Highcharts team has already done an excellent job in providing a comprehensive online reference, and each configuration is coupled with jsFiddle demos. This book also does not aim to be a chart design guide nor a tutorial on programming design with Highcharts.

In short, this book shows you what you can do with Highcharts.
What This Book Covers

The second edition includes four new chapters, a rewritten chapter, and new sections in some of the existing chapters. All the contents from the previous edition have been technically revised. As a result, the new edition consists of about 50 percent new material.

As this book contains a myriad of examples, it would be impractical to include all the source code of each example. For step-by-step tutorials, the code is listed incrementally. If you want to experiment with the sample code, you are strongly recommended to download the code from the Packt Publishing website or visit http://joekuan.org/Learning_Highcharts for online demos.

Chapter 1, Web Charts, describes how web charts have been done since the birth of HTML to the latest HTML5 standard with SVG and canvas technologies. This chapter also provides a short survey of charting software on the market using the HTML5 standard and discusses why Highcharts is a better product than others.

Chapter 2, Highcharts Configurations, covers the common configuration options in chart components with plenty of examples and explains how the chart layout works.

Chapter 3, Line, Area, and Scatter Charts, demonstrates plotting a simple line, area charts, and scatter charts to plotting a poster-like chart including all three series types.

Chapter 4, Bar and Column Charts, demonstrates bar and column charts as well as various derived charts such as a stacked chart, percentage chart, mirror chart, group chart, overlap chart, mirror stacked chart, and horizontal gauge chart.

Chapter 5, Pie Charts, demonstrates how to build various charts, from a simple pie chart to a multiseries chart, such as multiple pies in a chart and a concentric rings pie chart, that is, a donut chart. The chapter also explores how to create an open donut chart with specific options.

Chapter 6, Gauge, Polar, and Range Charts, is a step-by-step guide on constructing a twin dial speedometer and the creation of a simple solid gauge chart. It also demonstrates the polar chart's characteristics and its similarity to a Cartesian chart. It also illustrates the use of range data on area and column range charts.

Chapter 7, Bubble, Box Plot, and Error Bar Charts, explains the characteristics of bubble charts and their specific options. The chapter establishes a gradual approach on creating a bubble chart similar to a real life sport chart, applies the same exercise to a box plot with environmental data, and provides a tutorial on error bar charts using racing data.
Chapter 8, *Waterfall, Funnel, Pyramid, and Heatmap Charts*, illustrates how to configure waterfall and funnel charts and uses the drilldown feature to link both charts. Then there is a tutorial on pyramid charts and reconstructing them from a financial brochure. Then, heatmap charts are introduced and different outputs are shown by experiencing a number of series options.

Chapter 9, *3D Charts*, discusses what 3D charts in Highcharts really means and demonstrates the concept of 3D charts, such as column, pie, and scatter series. It illustrates specific 3D options with experiments and reconstructs a 3D chart from infographics. The chapter also covers how to create interactive 3D orientations with mouse actions.

Chapter 10, *Highcharts APIs*, explains the usage of Highcharts APIs and illustrates this by using a stock market demo to draw dynamic charts. The chapter discusses the use of different methods to update the series and analyzes the performance of each method on various browsers, as well as the scalability of Highcharts.

Chapter 11, *Highcharts Events*, explains Highcharts events and demonstrates them through various user interactions with charts from the portfolio application demos.

Chapter 12, *Highcharts and jQuery Mobile*, is a short tutorial on the jQuery Mobile framework and demonstrates how to integrate it with Highcharts by creating a mobile web application browsing an Olympic medals table. The chapter also covers the use of touch-based and rotate events with Highcharts.

Chapter 13, *Highcharts and Ext JS*, is a short introduction to Sencha's Ext JS and describes the components likely to be used in an application with Highcharts. It also shows how to use a module and a Highcharts extension in order to plot Highcharts graphs within an Ext JS application.

Chapter 14, *Server-side Highcharts*, discusses how Highcharts' server-side solution has evolved using PhantomJS. A quick introduction of PhantomJS is given and a step-by-step experiment is conducted to create a server-side solution using Highcharts. The chapter also demonstrates how to use the Highcharts official server-side script.

Chapter 15, *Highcharts Online Services and Plugins*, is a quick introduction to the export server service and discusses a recent significant cloud development: Highcharts Cloud. The chapter gives you a tour of what it offers and how to use it, from a basic level without any prior knowledge of Highcharts and JavaScript programming to an advanced user level. The chapter also demonstrates the structure of plugins and shows you how to combine multiple plugins to create a new user experience.
In this chapter, you will learn the general background of web charts. This includes a short history of how web charts used to be made before Ajax and HTML5 became the new standard. The recent advances in JavaScript programming will be briefly discussed. Then, SVG support and the new HTML5 feature canvas, the main drive behind JavaScript charts, are introduced and demonstrated. This is followed by a quick guide to the other JavaScript graphing packages that are available on the market. Finally, we are introduced to Highcharts and will explain the advantages of using Highcharts over other products. In this chapter, we will cover the following topics:

- A short history of web charting
- The rise of JavaScript and HTML5
- JavaScript charts on the market
- Why Highcharts?

**A short history of web charting**

Before diving into Highcharts, it is worth mentioning how web charts evolved from pure HTML with server-side technology to the current client side.

**HTML image map (server-side technology)**

This technique has been used since the early days of HTML, when server-side operations were the main drive. Charts were only HTML images generated from the web server. Before there were any server-side scripting languages such as PHP, one of the most common approaches was to use Common Gateway Interface (CGI), which executes plotting programs (such as gnuplot) to output images. Later, when PHP became popular, the GD graphic module was used for plotting. One product that uses this technique is JpGraph.
The following is an example of how to include a chart image in an HTML page:

```html
<img src="pie_chart.php" border=0 align="left">
```

The chart script file `pie_chart.php` is embedded in an HTML `img` tag. When the page is loaded, the browser sees the `img src` attribute and sends an HTTP request for `pie_chart.php`. As far as the web browser is concerned, it has no knowledge of whether the `.php` file is an image file or not. When the web server (with PHP support) receives the request, it recognizes the `.php` extension and executes the PHP scripts. The following is a cut down JpGraph example; the script outputs the image content and streams it back as an HTTP response, in the same way as normal image content would be sent back:

```php
// Create new graph
$graph = new Graph(350, 250);
// Add data points in array of x-axis and y-axis values
$p1 = new LinePlot($datay,$datax);
$graph->Add($p1);
// Output line chart in image format back to the client
$graph->Stroke();
```

Furthermore, this technology combines with an HTML `map` tag for chart navigation, so that, when users click on a certain area of a graph, for example a slice in a pie chart, it can load a new page with another graph.

This technology has the following advantages:

- Ideal for automation tasks, for example scheduled reports or e-mail alerts with the graph attached.
- Doesn't require JavaScript. It is robust, pure HTML, and is light on the client side.

It has the following disadvantages:

- More workload on the server side
- Pure HTML and a limited technology—few interactions can be put on the graphs and none on the animations
Java applet (client-side) and servlet (server-side)

A Java applet enables the web browser to execute multiplatform Java bytecode to achieve what HTML cannot do, such as graphics display, animations, and advanced user interactions. This was the first technology to extend traditional server-based work to the client side. To include a Java applet in an HTML page, HTML's applet (deprecated) or object tags are used and require a Java plugin to be installed for the browser.

The following is an example of including a Java applet inside an object tag. As Java does not run in the same environment in Internet Explorer as in other browsers, the conditional comments for IE were used:

```html
<!--[if !IE]> Non Internet Explorer way of loading applet -->
<object classid="Java:chart.class" type="application/x-java-applet"
height="300" width="550" >
<!--[endif] Internet way of loading applet -->
<object classid="clsid:8AD9C840..." codebase="/classes/">
<param name="code" value="chart.class" />
</object>
<!--[if !IE]> -->
</object>
<!--[endif]-->
```

Generally, the Java 2D chart products are built from the `java.awt.Graphics2D` class and the `java.awt.geom` package from Java Abstract Window Toolkit (AWT). Then, the main chart library allows the users to utilize it in a browser, extending from the `Applet` class, or to run it on the server side, extending from the `Servlet` class.

An example of a Java product is JFreeChart. It comes with 2D and 3D solutions and is free for nonprofit use. JFreeChart can be run as an applet, servlet, or standalone application. The following shows part of the code used to plot data points within an applet:

```java
public class AppletGraph extends JApplet {
    // Create X and Y axis plot dataset and populate
    // with data.
    XYPlot xyPlot = new XYPlot();
    xyPlot.setDataset(defaultXYDataset);
    CombinedDomainXYPlot combinedDomainXYPlot =
        new CombinedDomainXYPlot();
    combinedDomainXYPlot.add(xyPlot);
    // Create a jFreeChart object with the dataset
```
JFreeChart jFreeChart = new JFreeChart(combinedDomainXYPlot);
// Put the jFreeChart in a chartPanel
ChartPanel chartPanel = new ChartPanel(jFreeChart);
chartPanel.setPreferredSize(new Dimension(900, 600));
// Add the chart panel into the display
getContentPane().add(chartPanel);
}

To run a chart application on the server side, a servlet container is needed—for example Apache Tomcat. The standard web.xml file is defined to bind a URL to a servlet:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<web-app id="server_charts" version="2.4" xmlns="..." xmlns:xsi="..."
  xsi:schemaLocation="...">
  <servlet>
    <servlet-name>PieChartServlet</servlet-name>
    <servlet-class>charts.PieChartServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>PieChartServlet</servlet-name>
    <url-pattern>/servlets/piechart</url-pattern>
  </servlet-mapping>
</web-app>
```

When the servlet container, such as Tomcat, receives an HTTP request with the URL http://localhost/servlets/piechart, it resolves the request into a servlet application. The web server then executes the chart servlet, formats the output into an image, and returns the image content as an HTTP response.

This technology has the following advantages:

- Advanced graphics, animations, and user interfaces
- Reusable core code for different deployment options: client-side, server-side, or standalone applications

It has the following disadvantages:

- Applet security issues
- If the plugin crashes, it can hang or crash the browser
- Very CPU intensive
- Requires a Java plugin
- Long startup time
- Standardization problems
Adobe Shockwave Flash (client-side)

Flash is widely used because it offers audio, graphics, animation, and video capabilities on web browsers. Browsers are required to have the Adobe Flash Player plugin installed. As for plotting graphs, this technique was a common choice (because there weren't many other options) before the HTML5 standard became popular.

Graphing software adopting this technology ships with its own exported Shockwave Flash (SWF) files. These SWF files contain compressed vector-based graphics and compiled ActionScript instructions to create a chart. In order for the Flash Player to display the graphs, the SWF file has to be loaded from an HTML page. To do that, an HTML object tag is needed. The tag is internally created and injected into the document's DOM by the software's own JavaScript routines.

Inside this object tag is dimension and SWF path information for plotting the graph. The graph variable data is also passed inside this tag. So, as soon as the browser sees an object tag with specific parameters, it calls the installed Flash Player to process both the SWF file and the parameters. To pass the graph's plot data from the server side to the client-side Flash Player, flashVars is embedded inside a param tag with the data type. The following is an example from Yahoo YUI 2:

```html
<object id="yuiswf1" type="..." data="charts.swf" width="100%" height="100%">
  <param name="allowscriptaccess" value="always">
  <param name="flashVars" value="param1=value1&param2=value2">
</object>
```

This technology has the following advantages:

• Pretty graphics and animations with rich user interactions

It has the following disadvantages:

• If the plugin crashes, it can hang or crash the browser
• Very CPU-intensive
• Long startup time
• Standardization problems
The rise of JavaScript and HTML5

The role of JavaScript has shifted significantly from a few simple client routines to a dominant language to create and manage web user interfaces. The programming technique has moved towards object-oriented with the introduction of function object, prototype, and closure. This was driven by a group of pioneers such as Douglas Crockford, who was responsible for transforming the language to educate and make JavaScript a better language with his book *JavaScript: The Good Parts*, O’Reilly Media / Yahoo Press. Others include Sam Stephenson, creator of the Prototype JavaScript library (http://www.prototypejs.org), and John Resig, creator of the JQuery library (http://jquery.com), who brought JavaScript into a framework for building more complicated frontend web software.

It is beyond the scope of this book to give an introduction to this new programming style. Readers are expected to know the basics of jQuery and CSS selector syntax, which are used in some of the chapters. Readers should also be familiar with the advanced JavaScript scripting described in the book *JavaScript: The Good Parts*, O'Reilly Media / Yahoo Press, such as prototypes, closure, inheritance, and function objects.

HTML5 (SVG and Canvas)

In this section, two HTML5 technologies, SVG and Canvas, are covered, along with examples.

SVG

HTML5 is the biggest advance so far in the HTML standard. The adoption of the standard is growing fast (also fuelled by Apple mobile devices, which stopped supporting Adobe Flash). Again, it is beyond the scope of this book to cover them. However, the most relevant part to web charting is Scalable Vector Graphics (SVG). SVG is an XML format for describing vector-based graphics that is composed of components such as paths, text, shapes, color, and so on. The technology is similar to PostScript, except that PostScript is a stack-based language. As implied by its name, one of the major advantages of SVG is that it is a lossless technology (the same as PostScript): it doesn't suffer from any pixelation effects by enlarging the image. A reduced image size will not suffer from loss of original content.

Furthermore, SVG can be scripted with timing animation Synchronized Multimedia Integration Language (SMIL) and event handling. Apart from IE, this SVG technology is supported by all mainstream browsers, http://caniuse.com/#feat=svg-smil.
The following is a simple example of SVG code showing a single curved line between two points:

```xml
<svg xmlns="http://www.w3.org/2000/svg" version="1.1">
  <path id="curveAB" d="M 100 350 q 150 -300 300 0" stroke="blue" stroke-width="5" fill="none" />
  <!-- Mark relevant points -->
  <g stroke="black" stroke-width="3" fill="black">
    <circle id="pointA" cx="100" cy="350" r="3" />
    <circle id="pointB" cx="400" cy="350" r="3" />
  </g>
  <!-- Label the points -->
  <g font-size="30" font="sans-serif" fill="black" stroke="none" text-anchor="middle">
    <text x="100" y="350" dx="-30">A</text>
    <text x="400" y="350" dx="30">B</text>
  </g>
</svg>
```

The preceding SVG code is executed in the following steps:

1. Draw a path with id="curveAB" with data (d). First, move M to an absolute coordinate (100, 350), then draw a Bézier quadratic curve from the current position to (150, -300) and finish at (300, 0).
2. Group (g) the two circle elements—"pointA" and "pointB"—with the center coordinates (100, 350) and (400, 350) respectively with a radius of 3 pixels. Then fill both circles in black.
3. Group the two text elements A and B, started at (100, 350) and (400, 350), which display with the sans-serif font in black, and then shift along the x-axis (dx) 30 pixels left and right, respectively.

The following is the final graph from the SVG script:

![SVG graph](image)

The following is a simple example of SVG code showing a single curved line between two points:
Canvas

Canvas is another new HTML5 standard that is used by some JavaScript chart software packages. The purpose of Canvas is as its name implies; you declare a drawing area on the `canvas` tag, then use the new JavaScript APIs to draw lines and shapes in pixels. This is the distinct difference between the two techniques: Canvas is pixel-based, whereas SVG is vector-based. Canvas has no built-in animation routine, so API calls in timed sequences are used to simulate an animation. Also, there is no event-handling support, so developers need to manually attach event handlers to certain regions on the canvas. Fancy chart animation may prove more complicated to implement.

The following is an example of Canvas code that achieves the same effect as the preceding SVG curve:

```html
<canvas id="myCanvas" width="500" height="300" style="border:1px solid #d3d3d3;">Canvas tag not supported</canvas>
<script type="text/javascript">
    var c=document.getElementById("myCanvas");
    var ctx=c.getContext("2d");
    // Draw the quadratic curve from Point A to B
    ctx.beginPath();
    ctx.moveTo(100, 250);
    ctx.quadraticCurveTo(250, 0, 400, 250);
    ctx.strokeStyle="blue";
    ctx.lineWidth=5;
    ctx.stroke();
    // Draw a black circle attached to the start of the curve
    ctx.fillStyle="black";
    ctx.strokeStyle="black";
    ctx.lineWidth=3;
    ctx.beginPath();
    ctx.arc(100,250,3, 0, 2* Math.PI);
    ctx.stroke();
    ctx.fill();
    // Draw a black circle attached to the end of the curve
    ctx.beginPath();
    ctx.arc(400,250,3, 0, 2* Math.PI);
    ctx.stroke();
    ctx.fill();
    // Display 'A' and 'B' text next to the points
    ctx.font="30px 'sans-serif'";
    ctx.textAlign="center";
    ctx.fillText("A", 70, 250);
    ctx.fillText("B", 430, 250);
</script>
```
As you can see, both canvas and SVG can do the same task, but Canvas requires more instructions:

Instead of a continuous path description in SVG, a sequence of JavaScript drawing methods is called. The preceding Canvas code follows these steps to draw the curve:

- In the preceding example, it first calls `beginPath` to start a path in the canvas, then issues the `moveTo` call to move the path starting point to the coordinate (100, 250) without creating a line. The `quadraticCurveTo` method creates the curve path from the `moveTo` location to the top of the curve and the endpoint, which is (400, 250). Once the path is set up, we set up the stroke properties before calling the `stroke` method to draw the line along the path.
- We call `beginPath` to reset to a new path to draw and invoke the `arc` method to create a tiny circular path at both the start and end coordinates of the curve path. Once we have set up the stroke and fill styles, we call the `fill` and `stroke` routines to fill it black, to act as a coordinate marker.
- Finally, we set up the font properties and create the text A and B by calling `fillText` with the position near the coordinate markers.

In a nutshell, instead of a single tag with multiple attributes as in SVG, Canvas adopts multiple attribute-setting routines. SVG is mostly declarative, while Canvas enforces an imperative programming approach.

**JavaScript charts on the market**

It would be unimaginable to program a chart by hand-coding in SVG or Canvas. Fortunately, there are many different chart libraries on offer on the market. It is impossible to discuss each one of them.

We have omitted some of the products mentioned in the first edition of this book, due to a lack of competitive edge.
The chart libraries are open source, but some of them are short-lived in terms of not having a comprehensive set of basic charts, such as pie, line, and bar charts, and they look rather unfinished. Here, a handful of commercial and open source products are discussed, including all the basic charts and some with extras. Some of them still support the Flash plugin, which is an option for backwards-compatibility because SVG and canvas are not supported in older browsers. Although some of them are not free for commercial development, which is understandable, they are very affordable.

Many libraries use this add-on to emulate Canvas prior to IE 9.

**amCharts**

amCharts offers a full set of charts in both 2D and 3D, with other interesting charts such as radar, bubble, candlestick, and polar. All the charts look pretty and support animations. amCharts is free for commercial use, but a credit label will be displayed in the upper-left corner of the charts. The presentation of the charts looks nice and the animations are fully customizable. The default animations, compared to Highcharts, seems slightly overdone.

**Ext JS 5 charts**

Ext JS is a very popular Ajax application framework developed by Sencha, a pioneering company specializing in web application development. Ext JS 4 comes with a pure JavaScript charts library, unlike its predecessor Ext JS 3 that used the YUI 2 Flash chart library. As the market trend is moving away from Adobe Flash, Sencha responded with a home-brew charting library. Ext JS 4 covers all the basic 2D charts plus gauge and radar charts, and all the charts support animations. The license is free for open source and noncommercial usage, but a developer license is needed for commercial development. One great benefit of Ext JS charts is that integration with a comprehensive set of UI components, for example for a chart with a storage framework, makes displaying/updating both the chart and the table of data very simple to do with editors.

In Ext JS 5, the charting library has been completely restructured. Appearance is a major step forward compared to Ext JS 4: the charts look much more professionally done and on a par with other competitors. Although the Ext JS 5 chart layout, color, and animations may not be as stylish and smooth as Highcharts, it is a close call and still well-received.
Data Driven Documents

Data Driven Documents (D3) is the most widely-used charting library. It was created by Bostock et al., much improved from their prior academic research work on Protovis.


The principle of D3 is pretty unique, in that it focuses on transformation in document elements. In other words, it is a decorative framework to manipulate selected elements, known as selections, through a myriad of APIs. These APIs allow users to join the selections with chart data and style them like CSS or apply specific effects. This approach enables D3 to create a wide variety of impressive charts with animations that other products cannot produce. It is an ideal tool for plotting specific scientific graphs, or graphs that require complex data visualization presentation, such as Hierarchical Edge Bundling.

The software is free for commercial use and has attracted a large user base and contribution from user communities, especially from the academic sector.

Due to its highly programmable and comparatively low-level approach, it requires a much steeper learning curve that may not appeal to less technical users or developers looking for a chart-ready solution on a production level. As for constructing 3D charts in D3, it can be a challenging task. Although D3 trumps in performance, control, and presentation, it is not for everyone.

FusionCharts

FusionCharts is probably one of the most impressive-looking tools, and has the most comprehensive range of charts on the market. Not only does it come with a full variety of interesting 2D charts (radar, dial, map, and candlestick) available as a separate product, but it also offers fully interactive 3D charts. All the chart animations are very professionally done. FusionCharts can be run in two modes: Flash or JavaScript. Although FusionCharts comes with a higher price tag, it offers the best looking charts bundled with rotatable, animated 3D charts in column, pie, funnel, and pyramid series.
Raphaël

Raphaël is another free graphics library. It supports both SVG and VML for earlier browsers. It is designed like a generic graphics library, handling SVG elements, but it can be used as a charting solution. The software offers APIs to create a Canvas-like object, paper, that users can then create basic shapes or line SVG elements in. The API construct is somewhat similar to D3 in the sense that it can bind data into elements and manipulate them, but with rather more primitive methods, whereas D3 is a data-centric solution. The documentation is basic and it has a smaller user community. Like D3, it requires more effort to program a presentable chart than Highcharts and is not an ideal choice for 3D chart solutions.

Why Highcharts?

Highcharts offers very appealing and professional-looking 2D/3D charts on the market. It is a product that stands out by paying attention to details, not only on the presentation side, but also in other areas that are described later on. It was developed by a Norwegian company called Highsoft AS, created and founded by Torstein Hønsi, and released in late 2009. Highcharts is not their first product, but is by far their best-selling one.

Highcharts and JavaScript frameworks

Although Highcharts is built with the JavaScript framework library, it is implemented in such a way that it doesn't totally rely on one particular framework. Highcharts is packaged with adapters to make its interfaces to frameworks pluggable.

As a result, Highcharts can be incorporated with MooTools, Prototype, or jQuery JavaScript frameworks. Highcharts also has a standalone framework for those who write in pure JavaScript. This empowers users without compromising their already-developed product, or allows them to decide to use the framework that is best suited for their projects. Users who develop their web charting applications in jQuery are only required to load the jQuery library before Highcharts.

To use Highcharts in the MooTools environment, users simply do the following:

```html
<script src="http://ajax.googleapis.com/ajax/libs/mootools/1.4.5/mootools-yui-compressed.js"></script>
<script type="text/javascript"
  src="http://code.highcharts.com/adapters/mootools-adapter.js"></script>
<script type="text/javascript"
  src="http://code.highcharts.com/highcharts.js"></script>
```
To use Highcharts under Prototype, users need to do the following:

```html
<script src="http://ajax.googleapis.com/ajax/libs/prototype/1.7.1.0/prototype.js"></script>
<script type="text/javascript" src="http://code.highcharts.com/adapters/prototype-adapter.js"></script>
<script type="text/javascript" src="http://code.highcharts.com/highcharts.js"></script>
```

**Presentation**

Highcharts strikes the right balance of look and feel. The charts themselves are visually pleasant and yet the style is simple. The default choices of color are soothing without a sharp contrast and don’t conflict with each other, aided by the subtle shadow and white border effects. None of the text nor the colors of the axes are in black or any dark color, which keeps the viewer’s attention centered on the colored data presentation. The following is an example of a Highcharts representation:

![Highcharts example](image)

All the animations (initial, update, tool tip) in Highcharts are finely tuned: smooth with a gradual slowdown motion. The initial animation of the donut chart, which is a multi-series pie chart, is the most impressive one. This is the area in which Highcharts is clearly better. The animations in other charts are too mechanical, too much, and sometimes off-putting:
Web Charts

The round corners of tool tips and legends (both inner and outer) with a simple border do not fight for the viewer's attention and nicely blend into the chart. The following is a tool tip sample:

![Temperature Tool Tip Example](image)

70 km: -55.7°C

The following is a legend example with two series:

![Legend Example](image)

Male  Female

In a nutshell, each element in Highcharts does not compete with others for the viewer's attention, so they share the load equally and work together as a chart.

License

Highcharts has free noncommercial as well as commercial licenses. The free license for personal and nonprofit purposes is Creative Commons – Attribution Noncommercial 3.0. Highcharts offers different flavors of commercial license for different purposes. They have a one-off single website license and, thankfully, a developer license. For web development products, a developer license is a better model than charging in units of website use or a very high-priced OEM license because of the following reasons:

- It is easier for software companies to work out the math in their development plans
- There is less worry regarding how many copies are being sold, so as not to violate the license

As usual, a developer license does not automatically grant the use of Highcharts indefinitely. The license only permits the unlimited use of all the versions released within a year from the license purchase date. Thereafter, a brand new license is required if developers decide to use a newer version. Moreover, any condition can be negotiated for the OEM license, and the quote is usually based on the number of developers on the project and the number of deployments.
Simple API model

Highcharts has a very simple API model. To create a chart, the constructor API expects an object specifier with all the necessary settings. To dynamically update an existing chart, Highcharts comes with a small set of APIs. The configuration properties are described in detail in Chapter 2, Highcharts Configurations. The API calls are discussed in Chapter 10, Highcharts APIs.

Documentations

Highcharts' online documentation is one of the areas that really outshines the others. It is not just a simple documentation page that dumps all the definitions and examples. It's a documentation page built with thought.

The left-hand side of the documentation page is organized in an object structure, as you would pass it to create a chart. You can further expand and collapse the object's attributes as in a JavaScript console. This helps users to become familiar with the product by using it naturally:

```javascript
    series : [{
        data : "",
        dataParser : ,
        dataURL : null,
        legendIndex : undefined,
        name : "",
        stack : null,
        type : "line",
        xAxis : 0,
        yAxis : 0
    }],
```

The well-thought-out part of the documentation is on the right-hand side with the definitions of attributes. Each definition comes with a description and an online demonstration for each setting, linking to the jsFiddle website:

```
<table>
<thead>
<tr>
<th>type : String</th>
</tr>
</thead>
<tbody>
<tr>
<td>The type of series. Can be one of area, areaspline, bar, column, line, pie, scatter or spline. Defaults to &quot;line&quot;.</td>
</tr>
<tr>
<td>Try it: Line and column in the same chart</td>
</tr>
</tbody>
</table>
```

This instant jsFiddle demo invites users to explore different property values and observe the effect on the chart, so the whole documentation-browsing process becomes very effective and fluid.
Openness (feature request with user voice)

One important way that Highcharts decides new features for every major release is via the users' voice (this is not unusual in open source projects, but it is one of the areas in which Highcharts is better than others). Users can submit new feature requests and then vote for them. The company then reviews the feature requests with the most votes and draws up a development plan that includes the new features. The details of the plan are then published on the Highcharts website.

In addition, Highcharts is hosted on GitHub, an online public source control service, which allows JavaScript developers to contribute and clone their own versions:

<table>
<thead>
<tr>
<th>721 votes</th>
<th>3D charts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add new 3D style pie and bar charts, like <a href="http://www.jfree.org/jfreechart/images/PieChart3DDemo1.png">http://www.jfree.org/jfreechart/images/PieChart3DDemo1.png</a></td>
<td></td>
</tr>
<tr>
<td>64 comments</td>
<td></td>
</tr>
</tbody>
</table>

Highcharts – a quick tutorial

In this section, you will see how to implement your first Highcharts graph. Assuming that we want to run Highcharts from our local web server, first download the latest version from the Highcharts website: http://www.highcharts.com/download.

Alternatively, we can load the library via the Content Delivery Network (CDN), as follows:

```html
<script type="text/javascript"
    src="http://code.highcharts.com/highcharts.js"></script>
```

For debugging, highcharts.src.js is also available from the CDN.

Directory structure

When you unpack the downloaded ZIP file, you should see the following directory structure under the Highcharts-4.x.x top-level directory:
The following is what each directory contains and is used for:

- **index.html**: This is the demo HTML page, which is the same as the demo page on the Highcharts website, so that you can experiment with Highcharts offline.

- **examples**: This contains all the source files for the examples.

- **graphics**: This contains image files used by the examples.

- **gfx**: This contains an image file required to produce the radial gradient effect for browsers that only support VML.

- **exporting-server**: This directory contains three main components: a Java implementation of the online export server, a PhantomJs toolkit to run Highcharts on the service side, and server script to service the chart export function using Batik. This directory is useful for users who need to set up their own internal export service or run Highcharts as a server process. See Chapter 14, Server-side Highcharts and Chapter 15, Highcharts Online Services and Plugin.

- **js**: This is the main directory for Highcharts code. Each JavaScript filename has two suffixes: .src.js, which contains the source code with comments in it, and .js, which is the minified version of the JavaScript source file. The highcharts.js file contains the core functionality and the basic chart implementations, highcharts-3d.js is the 3D charts extension, and highcharts.more.js contains additional series such as polar, gauge, bubble, range, waterfall, errobar, and boxplot. Highcharts-all.js lets users provide all the chart series in their application.
• adapters: This contains the default adapter standalone-framework in source and compressed format.

• modules: Since the last edition, Highcharts has created a number of plugin modules such as solid-gauge, funnel, exporting, drilldown, canvas-tools and so on. This directory contains these plugins.

A third-party tool, canvg, supports Android 2.x, as the native browser has no SVG support but can display the canvas.

• themes: This has a set of JavaScript files prebuilt with settings such as background colors, font styles, axis layouts, and so on. Users can load one of these files in their charts for different styles.

All you need to do is move the top-level Highcharts-4.x.x/js directory inside your web server document's root directory.

To use Highcharts, you need to include Highcharts-4.x.x/js/highcharts.js library in your HTML file. The following is an example showing the percentage of web browser usage for a public website. The example uses the minimal configuration settings for getting you started quickly. The following is the top half of the example:

```html
<!DOCTYPE HTML>
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
  <title>Highcharts First Example</title>
  <script src="http://ajax.googleapis.com/ajax/libs/jquery/1.8.2/jquery.min.js"></script>
  <script type="text/javascript" src="Highcharts-4.0.4/js/highcharts.js"></script>
  We use the Google public library service to load the jQuery library version 1.8.2 before loading the Highcharts library. At the time of writing, the latest jQuery version is 2.1.1 and the Highcharts system requirement for jQuery is 1.8.2.

The second half of the example is the main Highcharts code, as follows:

```
Chapter 1

{ type: 'spline'
  title: {
    text: 'Web browsers statistics'
  },
  subtitle: {
    text: 'From 2008 to present'
  },
  xAxis: {
    categories: [ 'Jan 2008', 'Feb', .... ],
    tickInterval: 3
  },
  yAxis: {
    title: {
      text: 'Percentage %'
    },
    min: 0
  },
  plotOptions: {
    series: {
      lineWidth: 2
    }
  },
  series: [{
    name: 'Internet Explorer',
    data: [54.7, 54.7, 53.9, 54.8, 54.4, ... ]
  }, {
    name: 'FireFox',
    data: [36.4, 36.5, 37.0, 39.1, 39.8, ... ]
  }, {
    // Chrome started until late 2008
    name: 'Chrome',
    data: [null, null, null, null, null, null,
      null, 3.1, 3.0, 3.1, 3.6, ... ]
  }, {
    name: 'Safari',
    data: [1.9, 2.0, 2.1, 2.2, 2.4, 2.6, ... ]
  }, {
    name: 'Opera',
    data: [1.4, 1.4, 1.4, 1.4, 1.5, 1.7, ... ]
  ]
})
</script>
The spline graph is created via an object specifier that contains all the properties and series data required. Once the chart object is created, the graph is displayed in the browser. Within this object specifier, there are major components corresponding to the structure of the chart:

```javascript
var chart = new HighCharts.Chart({
    chart: {
        ...
        title: '...',
        ...
    },
});
```

The renderTo option instructs Highcharts to display the graph onto the HTML <div> element with 'container' as the ID value, which is defined in the HTML <body> section. The type option is set to the default presentation type as 'spline' for any series data, as follows:

```javascript
chart: {
    renderTo: 'container',
    type: 'spline'
}
```

Next is to set title and subtitle, which appear in the center at the top of the chart:

```javascript
title: {
    text: 'Web browsers ... '},
subtitle: {
    text: 'From 2008 to present'
},
```

The categories option in the xAxis property contains an array of x-axis labels for each data point. Since the graph has at least 50 data points, printing each x-axis label will make the text overlap. Rotating the labels still results in the axis looking very packed. The best compromise is to print every third label, (tickIntervals: 3), which makes the labels nicely spaced out from each other.
For the sake of simplicity, we use 50 entries in `xAxis.categories` to represent the time. However, we will see a more optimal and logical way to display date and time data in the next chapter:

```javascript
xAxis: {
    categories: ['Jan 2008', 'Feb', .... ],
    tickInterval: 3
},
```

The options in `yAxis` are to assign the title of the y-axis and set the minimum possible value to zero; otherwise, Highcharts will display a negative percentage range along the y-axis, which is not wanted for this data set:

```javascript
yAxis: {
    title: {
        text: 'Percentage %'
    },
    min: 0
},
```

The `plotOptions` property is to control how each series is displayed, according to its type (line, pie, bar, and so on). The `plotOptions.series` option is the general configuration applied to all the series types instead of defining each setting inside the `series` array. In this example, the default `linewidth` for every series is set to 2-pixels wide, as follows:

```javascript
plotOptions: {
    series: {
        lineWidth: 2
    }
},
```

The `series` property is the heart of the whole configuration object that defines all the series data. It is an array of the `series` object. The `series` object can be specified in multiple ways. For this example, the name is the name of the series that appears in the chart legend and tool tip. The data is an array of y-axis values that have the same length as the `xAxis.categories` array, to form (x,y) data points:

```javascript
series: [{
    name: 'Internet Explorer',
    data: [54.7, 54.7, 53.9, 54.8, 54.4, ... ]
}, {
    name: 'FireFox',
    data: [36.4, 36.5, 37.0, 39.1, 39.8, ... ]
}, {
```
The following screenshot shows how the final Highcharts should look on the Safari browser:
The following screenshot shows how it should look on the Internet Explorer 11 browser:

The following screenshot shows how it should look on the Chrome browser:
The following screenshot shows how it should look on the Firefox browser:

![Highcharts First Example](image.jpg)

**Summary**

Web charting has been around since the early days of HTML, emerging from server-side technology to the client side. During this period, several solutions were adopted to work around the shortcomings of HTML. Now, with HTML5, which is rich in features, web charting has come back to HTML and this time it is for good, with the aid of JavaScript.

A number of JavaScript chart products were mentioned in this chapter. Among these, Highcharts emerged with a distinct graphical style and smooth user interactions. We also went through a simple chart example to experience how quick and easy it is to create our very first Highcharts chart.

In the next chapter, we will explore the Highcharts configuration object in greater detail with plenty more examples. The configuration object is the core part of the product that the structure serves as the common prototype for all the charts.
Where to buy this book

You can buy Learning Highcharts 4 from the Packt Publishing website.

Alternatively, you can buy the book from Amazon, BN.com, Computer Manuals and most internet book retailers.

Click here for ordering and shipping details.